Best Practice Questions on String topic in JAVA

Problem 1: Longest Palindromic Substring

Description:

Given a string s, find the longest palindromic substring in s.

Input Format:

1. A string s.

Output Format:

1. The longest palindromic substring.

Example:

Input:

babad

Output:

bab

Problem 2: Minimum Window Substring

Description:

Given two strings s and t, find the minimum window in s that contains all characters of t. If there is no such window, return an empty string.

Input Format:

1. A string s.
2. A string t.

Output Format:

1. The minimum window substring.

Example:

Input:

ADOBECODEBANCABC

Output:

BANC

Problem 3: Anagrams in a String

Description:

Given two strings s1 and s2, check if s2 contains any permutation of s1.

Input Format:

1. A string s1.
2. A string s2.

Output Format:

1. True if s2 contains any permutation of s1, otherwise False.

Example:

Input:

abeidbaooo

Output:

graphql
True

Problem 4: Longest Common Prefix

Description:

Write a function to find the longest common prefix string amongst an array of strings.

Input Format:

1. An array of strings strs.

Output Format:

1. The longest common prefix.

Example:

Input:

["flower","flow","flight"]

Output:

fl

Problem 5: Edit Distance

Description:

Given two strings word1 and word2, find the minimum number of operations required to convert word1 to word2. The operations are insertion, deletion, or replacement of a character.

Input Format:

1. A string word1.
2. A string word2.

Output Format:

1. The minimum number of operations.

Example:

Input:

intention
execution

Output:

5

Problem 6: Substring with Concatenation of All Words

Description:

You are given a string s and a list of words words. Find all starting indices of substring(s) in s that is a concatenation of each word in words exactly once and without any intervening characters.

Input Format:

1. A string s.
2. A list of strings words.

Output Format:

1. A list of starting indices.

Example:

Input:

barfoothefoobarman
["foo","bar"]

Output:

[0,9]

Problem 7: Valid Parentheses

Description:

Given a string s containing just the characters '(', ')', '{', '}', '[', ']', determine if the input string is valid. An input string is valid if the brackets are closed in the correct order.

Input Format:

1. A string s.

Output Format:

1. True if the string is valid, otherwise False.

Example:

Input:

()

Output:

True

Problem 8: Count and Say

Description:

The count and say sequence is a sequence of digits defined by the recurrence relation:

- countAndSay(1) is "1".
- To generate the next term in the sequence, read the digits of the current term, counting the number of digits in groups of the same digit.

For example, the sequence is: 1, 11, 21, 1211, 111221, ...

Input Format:

1. An integer n representing the n-th term in the sequence.

Output Format:

1. The n-th term in the count-and-say sequence.

Example:

Input:

4

Output:

1211

Problem 9: Group Anagrams

Description:

Given an array of strings, group the anagrams together. You may return the answer in any order.

Input Format:

1. An array of strings strs.

Output Format:

1. A list of lists of grouped anagrams.

Example:

Input:

["eat","tea","tan","ate","nat","bat"]

Output:

[["bat"],["nat","tan"],["ate","eat","tea"]]

Problem 10: Find the Repeated DNA Sequences

Description:

Given a string s that represents a DNA sequence, find all the sequences of length 10 that occur more than once in a DNA molecule.

Input Format:

1. A string s representing the DNA sequence.

Output Format:

1. A list of repeated sequences of length 10.

Example:

Input:

AAAAACCCCCAAAAACCCCCCAAAAAGGGTTT

Output:

["AAAAACCCCC","CCCCCAAAAA"]

Test Cases

Problem 1 Test Cases:

1. **Input:** babad **Output:** bab or aba
2. **Input:** cbbd **Output:** bb
3. **Input:** a **Output:** a
4. **Input:** ac **Output:** a or c
5. **Input:** racecar **Output:** racecar
6. **Input:** abcd **Output:** a or b or c or d
7. **Input:** noon **Output:** noon
8. **Input:** abcba **Output:** abcba
9. **Input:** abacdfgdcaba **Output:** aba
10. **Input:** level **Output:** level

Problem 2 Test Cases:

1. **Input:** ADOBECODEBANC, ABC **Output:** BANC
2. **Input:** a, a **Output:** a
3. **Input:** a, b **Output:** ``
4. **Input:** aabbcc, abc **Output:** aabbc
5. **Input:** ABCD, AB **Output:** AB
6. **Input:** aa, aa **Output:** aa
7. **Input:** aa, aaa **Output:** ``
8. **Input:** abcde, xyz **Output:** ``
9. **Input:** aaabbbccc, abc **Output:** aaabbbccc
10. **Input:** aabbcc, cba **Output:** bcb

Problem 3 Test Cases:

1. **Input:** ab, eidbaooo **Output:** True
2. **Input:** ab, eidboaoo **Output:** False
3. **Input:** abc, abcabc **Output:** True
4. **Input:** abcd, abcabcabc **Output:** False
5. **Input:** xy, xyzxyz **Output:** True
6. **Input:** xyz, xyzxyz **Output:** True
7. **Input:** abc, zxyzxyz **Output:** False
8. **Input:** abcd, abcdabcd **Output:** True
9. **Input:** ab, abac **Output:** True
10. **Input:** a, aabbcc **Output:** True

Problem 4 Test Cases:

1. **Input:** ["flower","flow","flight"] **Output:** fl
2. **Input:** ["dog","racecar","car"] **Output:** ``
3. **Input:** ["dog","dogg","doggy"] **Output:** dog
4. **Input:** ["a","b","c"] **Output:** ``
5. **Input:** ["abcdef","abcf","abc"] **Output:** ab
6. **Input:** ["a"] **Output:** a
7. **Input:** ["ab","abc","abcd"] **Output:** ab
8. **Input:** ["abcd","abce","abcf"] **Output:** abc
9. **Input:** ["abcd","bcda","cdab","dabc"] **Output:** ``
10. **Input:** ["xyz","xy","x"] **Output:** x

Problem 5 Test Cases:

1. **Input:** intention, execution **Output:** 5
2. **Input:** kitten, sitting **Output:** 3
3. **Input:** flaw, lawn **Output:** 2
4. **Input:** sunday, saturday **Output:** 3
5. **Input:** horse, ros **Output:** 3
6. **Input:** algorithm, algorithms **Output:** 1
7. **Input:** ab, ac **Output:** 1
8. **Input:** ab, bc **Output:** 2
9. **Input:** abcd, dcba **Output:** 4
10. **Input:** a, b **Output:** 1

Problem 6 Test Cases:

1. **Input:** barfoothefoobarman, ["foo","bar"] **Output:** [0,9]
2. **Input:** wordgoodgoodgoodbestword, ["word","good","best","word"] **Output:** [8]
3. **Input:** lingmindraboo, ["foo","bar","the"] **Output:** [0]
4. **Input:** abcdef, ["a","b","c"] **Output:** [0,1,2]
5. **Input:** abcdabcdabcd, ["abcd"] **Output:** [0,4,8]
6. **Input:** aabbcc, ["a","b","c"] **Output:** [0,2,4]
7. **Input:** aaa, ["a","aa"] **Output:** [0,1]
8. **Input:** abcdab, ["ab","bc"] **Output:** [0,1]
9. **Input:** catcatcat, ["cat"] **Output:** [0,3,6]
10. **Input:** ababc, ["abc","ab"] **Output:** [1,0]

Problem 7 Test Cases:

1. **Input:** () **Output:** True
2. **Input:** ([)] **Output:** False
3. **Input:** ([]) **Output:** True
4. **Input:** {[()]} **Output:** True
5. **Input:** ((({})) **Output:** False
6. **Input:** []{}() **Output:** True
7. **Input:** ((([[[{{}}]]])) **Output:** False
8. **Input:** ([]){} **Output:** True
9. **Input:** ((()) **Output:** False
10. **Input:** {[(])} **Output:** False

Problem 8 Test Cases:

1. **Input:** 1 **Output:** 1
2. **Input:** 2 **Output:** 11
3. **Input:** 4 **Output:** 1211
4. **Input:** 5 **Output:** 111221
5. **Input:** 6 **Output:** 312211
6. **Input:** 7 **Output:** 13112221
7. **Input:** 8 **Output:** 1113213211
8. **Input:** 3 **Output:** 21
9. **Input:** 9 **Output:** 31131122211311123113111213211331121113122122211 2
10. **Input:** 10 **Output:** 13211311123113112211

Problem 9 Test Cases:

1. **Input:** ["eat","tea","tan","ate","nat","bat"] **Output:** [["eat","tea","ate"],["tan","nat"],["bat"]]
2. **Input:** ["a","b","c"] **Output:** [["a"],["b"],["c"]]
3. **Input:** ["abcd","dcba","abdc","badc"] **Output:** [["abcd"],["dcba"],["abdc","badc"]]
4. **Input:** ["aaa","aaa","aaa"] **Output:** [["aaa","aaa","aaa"]]
5. **Input:** ["cat","act","dog","god"] **Output:** [["cat","act"],["dog","god"]]
6. **Input:** ["bb","bb","bb","cc"] **Output:** [["bb","bb","bb"],["cc"]]
7. **Input:** ["x","y","x","x","y","y"] **Output:** [["x","x","x"],["y","y","y"]]
8. **Input:** ["ab","ba","ab","ab"] **Output:** [["ab","ab","ab"],["ba"]]
9. **Input:** ["xyz","zyx","yxz"] **Output:** [["xyz","zyx","yxz"]]
10. **Input:** ["flower","flow","flight"] **Output:** [["flower","flow"],["flight"]]

Problem 10 Test Cases:

1. **Input:** AAAAACCCCCAAAAACCCCCCAAAAAGGGTTT **Output:** ["AAAAACCCCC","CCCCCAAAAA"]
2. **Input:** AAAAAAAAAA **Output:** ["AAAAAAAAAA"]
3. **Input:** TTTTTTTTTTTTTTTTTTTT **Output:** ["TTTTTTTTTT"]
4. **Input:** AGCTAGCTAGCTAGCT **Output:** ["AGCTAGCTAG"]
5. **Input:** GCGCGCGCGCGCGCGCGCGCG **Output:** ["GCGCGCGCGC"]

6. **Input:** ACGTACGTACGTACGT **Output:** ["ACGTACGTAC"]
7. **Input:** CGTAGCTAGC **Output:** ["CGTAGCTAGC"]
8. **Input:** AAAAABBBBCCCCCCDDD **Output:** ["AAAAABBBB"]
9. **Input:** ACGACGACGACG **Output:** ["ACGACGACG"]
10. **Input:** GATTACATGATTACAGATTACA **Output:** ["GATTACAGATT"]